2

Special Report
SEI-90-SR-6

Carnegie-Mellon University
Software Engineering Institute

DARK Technology Transition Plan

Judy Bamberger

April 1990

Special Report
SEI-90-SR-6
April 1990

DARK Technology Transition Plan

Judy Bamberger
DARK Transition Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

[illegible]
Karl H. Shingler
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

Table of Contents

List of Figures

DARK Technology Transition Plan

Abstract: The DARK Transition Project Plan is aimed at the transition of the concepts, models, and prototype implementation developed by the DARK development project over the past two years. This document:

1. Presents the background, rationale, and conceptual goals of the DARK Project.
2. Provides an overview of the phase approach to Technology Transition used within the SEI and its relationship to DARK Project activities.
3. Describes the goals and objectives of the DARK Transition Project in the context of the phase approach.
4. Defines the tasks constituting the DARK Transition Project.

Acknowledgements

Sincere thanks to those who participated in the DARK development project [Robert Firth (Project Leader), Currie Colket (Navy Affiliate), Dave Stinchcomb (Westinghouse Affiliate), Roger Van Scoy, Tim Coddington, Daniel Klein, and Lorraine Nemeth], Technology Transition personnel (assistance with the phase approach model), the numerous internal and external reviewers of the DARK Project, and the several internal and external reviewers of the DARK Transition Plan, including, most notably, Phil Thornley (Ferranti).

1. Introduction

The DARK Transition Project Plan is aimed at the transition of the concepts, models, and prototype implementation developed by the DARK development project over the past two years. The Project Plan for the DARK Transition Project:

- Defines success criteria for DARK transition.
- Demonstrates that, via the DARK Project (i.e., the definition and implementation of the Kernel), many of those success criteria were met.
- Indicates those criteria that are beyond the scope of a DARK-specific transition project.
- Defines those activities yet to be performed to complete DARK-specific transition.
- Proposes the reevaluation of the DARK Transition Project by 31 July 1990, including the identification of a clear follow-on direction being determined by that date. If no such direction can be determined, the DARK Transition Project will terminate, and the SEI will just respond to requests for information.

The original goal of the DARK Project was to provide a set of artifacts, documents and code, to: "... demonstrate that it is possible to develop application code entirely in Ada that has acceptable quality and real-time performance ... in a manner that avoids or mitigates the efficiency and maturity problems found in current Ada runtime implementations." [4] To *demonstrate*, as required above, the transition activity had to occur – that is, individuals, groups, and organisations had to review DARK *documents and code to become convinced that* DARK did (or did not) provide that capability. The DARK Project was *not* out to develop any radically new technological breakthroughs, but to demonstrate that well-understood, familiar, and reliable methods of building real-time, distributed systems could be done in Ada as well as in any programming language. KP

Thus, simply listed, the aims of the DARK Project, and the DARK Transition Project, are to:

1. Define requirements for a Kernel to support distributed, real-time Ada applications which are accepted as realistic by builders of real-time systems.
2. Build it to good standards, following good software engineering practices.
3. Test it to ensure a low level of residual errors.
4. Port it successfully to a significantly different target.
5. Document it to standards similar to those that must be used by our target audience.
6. Advertise it widely to ensure that information about DARK is available to all interested industry, academic, and government parties.
7. Give it away to a sufficiently large number of organisations so that substantial feedback and technical interchange is generated based upon experience of DARK (concepts, models, software) in practice.
8. Demonstrate it with an application accepted as realistic by builders of real-time systems.
9. Promote communication and information flow between all interested parties so that active support from the DARK project is no longer needed in continuing the development and enhancement of the DARK software.

10. Reevaluate the DARK transition activities on the basis of need, interest, and availability of self-sustaining communication and information flow.

During the DARK Project, the Kernel was defined [4], built (the code), tested (internal system test plans and system test cases, unit testing), ported (from a 68020 target to a VAX/VMS target), documented (see section 1.2), advertised (numerous "road shows" to industry, government, academia; presentations at conferences; publications in proceedings), given away (to nearly 20 Acceptor Sites).

What remains to be done, very simply, is to document what the current status is (e.g., who is doing what to which artifact) and then address the remaining three points. The first of these three, "demonstrate it ...," requires a commitment of technical people within the SEI – or a technical person within the SEI and one from a DARK user. As such, this is the riskiest of the remaining activities.

This sounds, perhaps, very self-defining: create those aims – after-the-fact, no less – such that the DARK Transition Project can fulfill them, whatever they may be. However, this is not the tack that is taken. The aims are analysed in the context of the "phase approach to transition," which is reviewed in Chapter 2.

The organisation of this plan is as follows:

Chapter 1	This chapter, which concludes with a presentation of the DARK Project background, rationale, and conceptual goals. This chapter also presents the current status of the DARK artifacts, the degree to which the Kernel software itself has been tested, and a list of SEI documents and other publications produced by the DARK project as of this writing.
Chapter 2	Provides an overview of the phase approach to Technology Transition, which is used within the SEI. The relationship of this approach to the DARK Project is summarised after this presentation. This chapter concludes with a synopsis of feedback from DARK External Contacts, both those that have the code and those having just the documentation.
Chapter 3	Describes the goals and objectives of the DARK Transition Project, in the context of the previous chapter. The goals of the DARK Project and its transition and the strategy for achieving them are described; the target audience is introduced and the mechanism by which closer customer/SEI relationships are to be formalised is described; and the success criteria against which the DARK Transition Project is to be measured are enumerated.
Chapter 4	Defines the tasks constituting the DARK Transition Project, including the priority and status of each task and the resources required by each.
Chapter 5	A list of references cited in this document.

1.1. Background

Ada is now being mandated for a large number of DoD development projects as the sole programming language to be used for developing software. Many of these projects are trying to build distributed real-time systems. Many project managers and contractors are anxious to support this effort, to reap the advantages of Ada, and to use the newer techniques of software engineering that Ada can support. This transition, however, has not always been smooth; some serious problems have been encountered.

1.1.1. Ada Runtime Environment

One of the most persistent and worrying problems is the suitability of the Ada runtime system, most notably the tasking features, and especially on distributed systems. There are issues concerning functionality (amply documented in [2]), customisation, tool support (especially target debuggers and performance monitors); issues of inter-process communication and code distribution; and, perhaps most intractable, issues of execution-time efficiency.

One way of approaching this problem is to press for better, "more mature" Ada implementations: more optimisation; user-tailorable runtime systems (as in [1]); special-purpose hardware. This is a valid route, but one that will take time, money, and experience, and many of the solutions will be compiler dependent, machine dependent, or application dependent. Many developers are still unsure even how to use the new language features of Ada, and at least one cycle of application use, performance measurement, and methodology review will be needed before users can be sure which parts of the Ada language and runtime are indeed critical.

The Kernel produced by the Distributed Ada Real-Time Kernel (DARK) project implements another route to a possible solution (defined at length in [4]) pursued at the Software Engineering Institute (SEI). It should be a quicker and cheaper route, and hence a feasible short-term alternative.

However, should the models on which the Kernel is built and the abstractions it exports continue to prove their efficacy in the user community, it is quite possible that the Kernel, or a Kernel-derived solution, may provide a long-term solution to a number of issues dealing with Ada in distributed, real-time systems.

1.1.2. Application and System Code

In conventional programming, application code (which is what has to be written to meet the user requirements) is distinguished from system code (which is obtained with the target machine, and which is intended to support applications generally). With Ada and embedded systems, these distinctions are not so clear cut. First, it has been traditional, when developing real-time systems in other programming languages, for the application programmer to write specific code down to a far lower level, including special device drivers, special message or signaling systems, and even a custom executive. There is far less general-purpose system code. Secondly, the Ada language complicates the distinction between application and system code. In older languages, almost all system functions were invoked through a simple and well-understood interface — the system call — expressed as a normal subroutine call. In Ada, however, many traditionally system-level functions are explicit in the language itself, or implied by language constructs; for example, *tasking, task communication, interrupt acquisition,* and *error handling.* In fact, the work is really done by the old familiar system code, now disguised as the *Ada runtime.*

1.1.3. Abstractions and Their Breakdown

If the user is satisfied with the Ada level of abstraction — with its view of what tasks are, what time is, and so on — then the Ada view is a simplification: the application code in fact performs system calls, but the compiler inserts them automatically as part of the implementation of language constructs.

Unfortunately, many users are dissatisfied with the Ada abstraction, and seek either finer control or access to lower-level concepts, such as semaphores, send/wait or suspend/resume primitives, and bounded delays. Under the above circumstances, the extra language features, and the hidden system calls they generate, are an active hindrance to the application programmer, and an obstruction to the work of implementation.

This, of course, is a distraction from the real work — the work of implementing the application. *One of the main motivators of the Kernel is the observation that many contractors using Ada are* spending most of their time worrying about the Ada system level and far too little time solving the application problems, some of which are not easy.

In sum, it can be harder to build applications using Ada language features than it would be to implement the required functionality without them. But it is also undesirable for every application to reinvent specific incarnations of real-time functional abstractions.

1.1.4. Distributed Applications

A further and equally difficult problem is the issue of executing applications on a distributed target configuration. Good software development methods teach decomposition of large applications into functional units communicating through well-defined interfaces. The physical allocation of such units to individual processors in the target environment can be done in many ways, without impairing their functionality. Good design therefore requires that the specification of these functional units and interfaces be independent, as far as possible, of their physical distribution.

In a real-time system, this implies that the mechanisms by which units interact — to synchronise, communicate, schedule one another, or alert one another — should be uniform, regardless of whether the units are sited on the same processor or at some distance across a distributed network. If the implementation language is Ada, this leads to a requirement for *distributed Ada.*

Unfortunately, nearly all current commercially-available Ada implementations do not support this requirement. They implement the real-time mechanisms of the language only on individual or isolated processors, and provide no help with communication between processors, and hence between units on different machines. This situation leads to systems where Ada tasks communicate by different mechanisms, with different style, semantics and implementations, merely because the Ada tasks are local in one case, and remote in the other. Overall, there is a substantial loss of application clarity, maintainability, reconfigurability, and conceptual economy.

1.1.5. Real-Time Requirements

This brings us to the crux of the Kernel's rationale. Users — people who have to write application code — do not want language features: they want language functionality. In Ada, much of the real-time functionality is captured in the form of special features. This may well be (the) correct solution in the long term ([3]), since by making real-time operations explicit in the language, the compiler is permitted to apply its intelligence to their optimisation and verification. But in the short term, it is palpably not working: the users either cannot use, or do not know how to use, the given features to achieve the required functionality; the implementors of the language do not know how to satisfy the variety of needs of real-time applications; the vendors are unable to customise extensively validated implementations; and commercial support for distributed targets is rare, even as the need for such support is becoming endemic among application developers.

Accordingly, it is opportune to revert to the former method of providing functionality: by specific system software implemented as a set of library routines and invoked explicitly by the user. The Kernel has taken this approach.

1.1.6. Purpose and Intended Audience

The main purpose of the Distributed Ada Real-Time Kernel (DARK) Project was to demonstrate that it is possible to develop application code entirely in Ada that has acceptable quality and real-time performance. This purpose was achieved by providing a prototype artifact — a Kernel — that implements the necessary functionality required by real-time applications, but in a manner that avoids or mitigates the efficiency and maturity problems found in current Ada runtime implementations.

This prototype embodies a tool-kit approach to real-time systems, one that allows the user to build application-specific, real-time abstractions. This prototype was not intended to solve all the problems of embedded, real-time systems, nor is it the only solution to these problems. However, it is intended to be a solution where efficiency and speed are the primary motivation and, where warranted, functionality has sometimes been limited accordingly.

The Kernel provides one solution to the problem of using Ada in distributed, real-time, embedded applications — one that can readily be accomplished in the near term. The Kernel is truly "in the spirit of Ada" — that is, it uses the Ada language features (e.g., packages, subprograms) to provide needed adjunct capabilities. This alternative returns explicit control of scheduling to the application implementor and provides a uniform communication mechanism for supporting distributed systems.

Other difficult areas, such as fault tolerance and multi-level security, were not directly addressed in the Kernel definition. The primitives have been studied in light of these and other equally demanding issues, and are simple and flexible enough to accommodate future development in these areas.

The goal of the Kernel was to provide a viable paradigm of near-term support to a wide number of real-time embedded applications currently being required to use Ada for implementation. This Kernel is based on the belief that applications builders, not compiler vendors or language designers, best know the system-level behavior required for their programs; and that standardisation of such behavior should be provided via a library package interface under the control of the application implementor, *not* via modifications to the Ada language. The strategy embodied in the DARK Kernel provides that kind of support.

1.2. DARK Development Project Status

The code for Version 3.0 of the Kernel, both the 68020-targeted and the VAX/VMS-targeted versions, was baselined by the DARK Project in October 1989. The final documentation update was approved in December 1989.

The pertinent pieces of the Kernel environment are:

- The host system is a DEC VAX operating under VMS 5.0 (the specific host used at the SEI is a MicroVAX II operating under MicroVMS 5.0).
- The Kernel exists in two versions:
    1. An implementation in Ada using the TeleSoft Telegen2 V3.22 Ada Development System (of which, the OASYS XA68000 V4.12 Cross-Assembler is a part) and MC68020 assembler targeted to the network of MC68020 processors and peripherals described in the *Kernel Architecture Manual.*
    2. An implementation in Ada using the VAX Ada V1.5 and VAX Macro targeted to the VAX-11 architecture running VMS V5.0.

Compliance with the Kernel behaviour requirements (*Kernel Facilities Definition* (KFD) subsection .1 of Chapters 5 - 13) was determined through unit testing. The results of this are attached to the KFD. The Kernel meets nearly all of its behaviour requirements, as demonstrated via unit testing.

Overall, the Kernel does not meet most of the performance requirements (KFD subsection .2 of Chapters 5 - 13) that were set as initial goals at the initiation of the DARK development project – some by a factor of 2:1, others by an order of magnitude (base 10).

The KFD does contain actual performance numbers in Appendix D. Many of the measured values do not have the requisite time deductions applied to them; thus, the reported results measure more than the performance requirement states should be measured. For example, requirement 10.2.1, Transmission time of 0-length message, "shall be no more than 25 microseconds" specifically notes "excluding network transmission time." The value reported in KFD Appendix D, 2502.79 microseconds, does **not** account for that deduction.

All reported values reflect a Kernel to which no optimisations (either via the compiler or via Kernel code restructuring) have been applied.

The DARK Project defined and used a software development process that it documented in a software development plan, which included design, coding, and documentation style guides. While this process does not map directly into DoD-STD-2167(A), the requirements for design, coding, testing, configuration management, and more are similar to those found in the community the SEI is mandated to serve. The code and supporting documentation reflect their use. These guides are all internal SEI documents, used to guide the DARK Project; there is currently no plan to release any of them.[1]

The standard Kernel release package consists of:

- 1 Magnetic tape in 6250 bpi VMS Backup format
- 1 each of the following documents:
  - *Version Description and Installation Guide* (VDIG)
  - *Kernel Facilities Definition* (KFD)
  - *Kernei User's Manual, two volumes* (KUM)
  - *Kernel Architecture Manual* (KAM)
  - *Kernel Porting and Extension Guide* (PORT)

Documents have traditionally bee ; sent out upon request, pending the DARK Project Leader's concurrence where required.

The following documentation, reports, and papers were produced by DARK Project team members and are available (hard copy) upon request:

SEI reports describing the Kernel and its use.

[OVVW] Bamberger, J., Coddington, T., Colket, C., Firth, R., Klein, D., Stinchcomb, D., Van Scoy, R.
Distributed Ada Real-time Kernel.
Technical Report CMU/SEI-88-TR-17, ADA199482, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, updated 1989.

Summary: This document provides an overview of the design of the Distributed Ada Real-Time Kernel software artifact by describing:

- The rationale behind the Kernel development;
- The underlying models, assumptions, and restrictions that govern the design and implementation of the Kernel; and
- An overview of all capabilities provided by the Kernel to real-time, distributed Ada applications.

This document is geared toward: engineers who want an overview of the models and capabilities of the Kernel.

[KFD] Bamberger, J., Coddington, T., Colket, C., Firth, R., Klein, D., Stinchcomb, D., Van Scoy, R.
Kernel Facilities Definition.
Technical Report CMU/SEI-88-TR-16, ADA198933, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, updated 1989.

Summary: This document defines the conceptual design of the Distributed Ada Real-Time Kernel software artifact by specifying:

[1]However, the possibility of "product-ising" the DARK style guide is being re-evaluated.

- The rationale behind the requirements for the Kernel and the goals to be achieved by it;
- The underlying models, assumptions, and restrictions that govern the design and implementation of the Kernel; and
- The behavioral and performance requirements to which the Kernel is built.

This document is both the requirements and top level design document for the Kernel. This document is geared toward: project personnel who are analyzing the system models supported by the Kernel to determine its applicability for a particular domain, designers and engineers who want an overview of each Kernel primitive, and engineers who require a precise description of required Kernel behavior and performance.

[KAM] Bamberger, J., Coddington, T., Colket, C., Firth, R., Klein, D., Stinchcomb, D., Van Scoy, R.
Kernel Architecture Model.
Technical Report CMU/SEI-89-TR-19, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1989.

Summary: This document contains the detailed design description of the Kernel. The overall system architecture and the rationale for it are presented as relevant to both the application (i.e., the *external* view of the Kernel) and the Kernel maintainer (i.e., the *internal* view of the Kernel). This document presents the algorithms and data structures needed to implement the functionality defined in the *Kernel Facilities Definition*. This document also contains an in-depty description of the communication protocol used by the Kernel, both the network software hardware that compose the DARK testbed at the SEI, and a detailed enumeration of all compiler dependencies exploited by Kernel software. This document is geared toward engineers responsible for porting and maintaining the Kernel and engineers requiring detailed information about the internals of the Kernel.

[KUM] Bamberger, J., Coddington, T., Firth, R., Klein, D., Stinchcomb, D., Van Scoy, R.
Kernel Users Manual.
Technical Report CMU/SEI-89-UG-1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1989.

Summary: This document serves as both a traditional user's manual and as a reference manual. For the general user, this document describes: the models underlying the Kernel, its concept of operations, and the primitives available to the application program. For those requiring more insight into the working of and the resource usage of the Kernel, its internal data structures, scheduling algorithms, and execution-time resource consumption are also provided. Information to tailor and prepare the Kernel for application-specific use and target environments is enumerated. Also provided are the motivation and design for a number of abstractions typically required by embedded systems that may be built readily on top of Kernel primitives. The specification of the Kernel is provided as a separately bound appendix to the Kernel User's Manual.

[PORT] Bamberger, J., T. Coddington, R. Firth, D. Klein, D. Stinchcomb, R. Van Scoy.
DARK Porting and Extension Guide.
Technical Report CMU/SEI-89-TR-40, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1989.

Summary: This document describes two activities, both related to modifying the Kernel:

- It characterizes the changes needed to port the Kernel to another architecture. It does this by detailing the changes made to the Kernel when porting it from the bare-board 68020 target to the VAX/VMS host.
- It describes some enhancements that might be made to the Kernel to better support real-time applications.

[VDIG] Bamberger, J., Coddington, T., Firth, R., Klein, D., Stinchcomb, D., Van Scoy, R.
Version Description and Installation Guide.
Technical Report CMU/SEI-89-TR-20, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1989.

Summary: This document characterizes a specific version of the Distributed Ada Real-Time Kernel (DARK) software artifact and supplies documentation for its installation and use. This document is geared toward: the engineer responsible for installing the Kernel, engineers responsible for porting and maintaining the Kernel, and engineers using the Kernel and needing an awareness of changes from the previous release.

Other articles relating to the Kernel.

[NAE88] Bamberger, J. and Van Scoy, R.
Distributed Ada Real-Time Kernel.
In *Proceedings NAECON '88.* May, 1988.

Summary: This paper describes the Kernel in terms of the ISO seven layer Open Systems Interface model, concentrating on the technical aspects of process creation, scheduling and communication in a distributed environment.

[RTAW88] Bamberger, J. and Van Scoy, R.
Returning Control to the User (where it belongs).
Position paper presented at the 2nd International Workshop on Real Time Ada Issues, Devon, UK, June 1-3 1988.

Summary: This position paper describes one approach to using Ada for real-time applications. The dominant theme of this paper (and of the DARK Project as a whole) is that only the application implementor knows how the application should behave. Taking this as a given, the paper asserts that control belongs to the user, and hence steps must be taken to provide the user (i.e., application implementor) with all the means needed to properly build an application. The paper illustrates this via the design approach taken by the DARK Project and how the Kernel consistently allows the user to make the needed decisions. This paper was published in the *Proceedings of the Ada UK International Conference*, Pages 29 - 34, issued as a supplement to Volume 9 of *Ada User*, 1989.

[RTAW89] Bamberger, J., Firth, R., Van Scoy, R.
Considerations for Ada 9x Based upon Real Project Experience.
Position paper presented at the 3rd International Workshop on Real Time Ada Issues, Nemacolin Woods, PA, June 26-29, 1989, publication TBD.

Summary: This position paper describes the view held by some of the DARK project members on key requirements of real-time systems being procured and developed for today's embedded systems, the support that Ada does and does not provide to meet those system-level requirements, how the Kernel was designed and developed to address those requirements, and in light of the research performed and experience gained during that time, provides some recommended considerations for the Ada 9x language requirements and design efforts.

[NGCR] Van Scoy, R., J. Bamberger and R. Firth.
An Overview of DARK. In *Proceedings 1989 Workshop on Operating Systems for Mission Critical Computing,* Pages Y.1-Y.11. September 1989.

Summary: This paper presents a top-level view of the Kernel concepts and models.

[AdaLet89] Van Scoy, R., J. Bamberger and R. Firth.
An Overview of DARK. *Ada Letters*, Volume IX, Number 7, Pages 91-101. November/December, 1989.

Summary: This paper presents a top-level view of the Kernel concepts and models. It is the first part of a two-part article.

[AdaUK89] Van Scoy, R. and J. Bamberger
Reuse and Reality. In *Proceedings of the Ada UK International Conference*, Pages C132-C141, issued as a supplement to Volume 10 of Ada User, 1989.

Summary: This paper presents an overview of considerations and tradeoffs related to reuse during the requirements, design, and implementation phases, in the context of the development and testing of the DARK Kernel artifact.

2. Phase Approach

This chapter begins by setting the context for the DARK Transition Project by defining the phase approach to transition, which is used by the SEI Technology Applications organisation, to frame discussions about transition concepts. Following this, the activities of the DARK Project are mapped into this framework, demonstrating that transition activities have been occurring throughout the DARK Project. Some of these tasks were done with formal assistance from Transition personnel or with an eye to an underlying transition "model", some without; in any case, the transition was occurring. Lastly, feedback from those External Contacts who have received documentation and/or code or have attended DARK reviews and/or presentations is provided, confirming where the DARK Project is as far as meeting its goals and the goals of any DARK-specific transition activity.

2.1. Setting the Context

This chapter reviews the "phase approach to transition" (see Figure 2-1).[2]

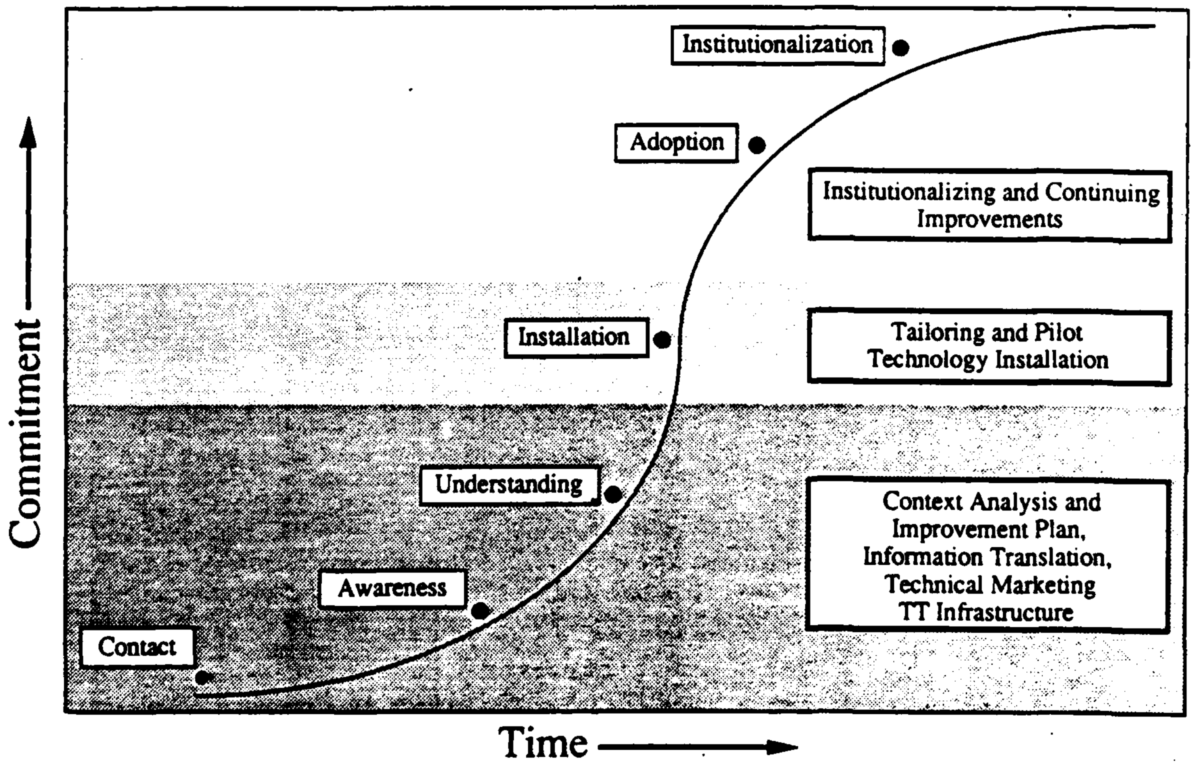
Institutionalization
Adoption
Institutionalizing and Continuing Improvements
Installation
Tailoring and Pilot Technology Installation
Understanding
Context Analysis and Improvement Plan, Information Translation, Technical Marketing TT Infrastructure
Awareness
Contact
Commitment
Time

Figure 2-1: Phase Approach to Transition

Technology transition is a phased activity that moves an *organisation* through a sequence of transition stages for a given *technology*.

[2]Adapted from words by Stan Przybylinski (SEI).

When attempting to introduce a new technology, target audiences for the transition[3] pass through different stages of commitment to the technology. The early stages focus on the acquisition and comprehension of information about the technology. As a target audience moves through the life cycle, the later stages focus on relative commitment to actual use of the technology. Transition is successful when the target audience reaches the stage of commitment appropriate for that technology in its organisation.

Each of the points on the curve in Figure 2-1 can be described as follows:

1. *Contact:* The target audience has had initial contact with the technology through some means (e.g., documents, briefings, marketing information); the target audience had not yet heard of the technology prior to this.
2. *Awareness:* That initial contact (or others) makes the target audience aware of the existence of the technology; the target audience now acknowledges the existence of that technology.
3. *Understanding:* The target audience understands the technology well enough to be conversant in the relevant details.
4. *Installation:* The target audience agrees to use the technology for some purpose on a trial basis (e.g., a pilot project, prototype development). This is often done in preparation for adoption of a technology.
5. *Adoption:* The target audience agrees to use the technology more widely within its organisation for an application related to its business purpose.
6. *Institutionalisation:* The use of the technology is made part of the standard practices and procedures of the organisation of the target audience.
7. *Internalisation:* The use of the technology becomes the way individuals within the organisation personally prefer to perform their work.[4]

The first three points on the graph, *contact*, *awareness*, and *understanding* constitute a *context analysis and improvement* phase. During this phase, an organisation typically assesses itself with the goal of defining the differences between what currently exists in practice and what must exist for the organisation to meet the desired improvement goals.

The next point on the graph, *installation*, constitutes a *technology tailoring and installation* phase, which requires the identification of groups by a client organisation to begin using the technology. These groups work on pilot studies in the use of the technology, identifying and solving difficulties arising from the studies, and they prepare an implementation plan for broader acceptance of the technology by the organisation.

The next two points on the graph, *adoption* and *institutionalisation*, constitute an *institutionalising and continuing improvement* phase, wherein the new technology is broadly implemented across appropriate parts of the client organisation. The focus here is on refining the technology and

[3]Target audiences can be organisational units of different levels (e.g., individuals, projects, programs, business units, etc.) depending on the level of analysis.

[4]Not shown in Figure 2-1, as this is a personal activity, one that could be occurring during any stage of organisational transition to a new technology.

solving difficulties with implementation across groups in the organisation.[5]

The goal for a particular transition effort is the introduction and use of the technology *to the extent relevant for that situation.* Based on these stages, it is easy to see why, for example, *internalisation* should not always be the goal. For technologies that are subject to change (e.g., specific software tools), *adoption* might be sufficient to make productive use of the technology. Note that, for those technologies where *commercialisation* would be a prerequisite for *institutionalisation* and *internalisation*, additional market forces come to bear, over and above the cultural and organisational forces more germane to the first phases.

These stages occur at different times for different organisations (or even for different parts of the same organisation). Given that the technology in question meets the known or latent needs of an organisation, technology transition is most effective when the right information is provided or developed in the earlier stages for use in "selling" the technology in subsequent stages. These information requirements differ widely in different contexts (e.g., a university researcher may need much less information to adopt a particular tool than might an industry practitioner that has a schedule to meet). In general, increasing the amount of relevant information available to any potential user increases the possibility of successful transition.

2.2. Relationship to the DARK Project

The DARK Project, both with and without the assistance of the Technology Transition Programme, found itself traversing many of the points on this curve with the External Contact community. The numbered paragraphs that follow step through the phases of technology transition (as presented in Chapter 2.1 and illustrated in Figure 2-1), giving a brief description of activities occurring in the general domain of real-time systems builders is provided, followed by a discussion of related activities performed by the DARK development project team.

1. *Contact.*

   The audience that the DARK Project targeted had already come into *contact* with the requirements for and several alternative solutions to the issue of dealing with: distribution, real-time, Ada, and embedded – as that was the community from which DARK derived its initial requirements.

   DARK Activity. As noted above, the DARK system requirements were derived from the community to which the results were to be transitioned. The DARK Project gathered inputs from this community during Project conceptualisation and requirements analysis. This interaction was effective at a number of levels:

   - The Project stayed in touch with the "user community." An extensive list of External Contacts was developed throughout the entire process, and a network of people/groups/organisations dealing with similar issues was built.
   - The Project stayed in touch with the "real world," avoiding the "ivory tower syndrome." (In some cases (e.g., ATF) this helped demonstrate that the SEI, through the DARK Project, was not afraid to deal with the *real and difficult*

[5]Adapted from *Technology Applications Plan*, DRAFT Version 2.6, 8 August 1989.

issues in software development; this was a positive turning point in the SEI relationship with the ATF SPO.)

- The Project demonstrated a willingness to be open to reviews and comments, setting an example for industry in this. (One CDR attendee later noted that it was delightfully unusual for government to open itself up to industry for critique for a change; that was the key motivator for his wanting and being able to attend the DARK CDR.)

2. *Awareness.*

The target community is already *aware* that there are issues of dealing with: distribution, real-time, Ada, and embedded. They are aware of, and may have developed, one or more solutions in support of Ada applications and non-Ada applications.

DARK Activity. The DARK Project identified models for many of the areas of concern to real-time system developers, for example: interrupt and time (relative and absolute) encapsulation, inter-processor communication, and an implementation strategy more in-tune with embedded computer system needs. DARK system models built on a more familiar "process" model versus the "new" (and Ada-specific) rendezvous concept. These encapsulations and abstractions were to be designed using packages and subprograms to extend the primitives and abstractions provided directly by the Ada programming language. The "motto" of the DARK Project became: *returning control to the user* – an echo of a continuing requirement heard from real systems developers.

The DARK Project members presented their high-level concepts in public fora, including "road shows" at: the SEI itself (the System Model Review [SMR]), DoD programme offices (e.g., ATF), and industry. The *Kernel Facilities Definition* was widely disseminated (internally and externally) for review and comment; invaluable comments and perspectives were received from real systems builders and were incorporated as appropriate.

3. *Understanding.*

The audience already *understood* there were issues about building real-time, distributed systems and using Ada, and were able to articulate one or more ways of addressing them – both through the primitives of the Ada language itself *where appropriate* and through lower-level kernels as required.

DARK Activity. DARK Project members refined the models and abstractions and implemented the Kernel. A Critical Design Review [CDR] was held; 40+ people attended, most of them from outside the SEI; most of those were real systems builders. During the CDR, the process and inter-process communication models were validated, the model for interrupt encapsulation was refined, and the abstraction of time was changed – in response to real system requirements. In addition, more "road shows" were given to industry, and the DARK architecture was presented at conferences, including AdaJUG, SIGAda, etc. Preliminary DARK reports were disseminated upon request, and the DARK Project provided (as yet paper-only) artifacts against which other organisations could evaluate their work-in-progress or procurements or plans, etc.

4. *Installation.*

During the time of the DARK Project, industry was developing its own solutions to the issues of real-time Ada kernels – both by building their own from scratch or working with compiler vendors. Some of these solutions were noticeably "DARK-like" (not surprisingly); the heavily tailored Ada compilers often provided very "DARK-like" interfaces (also not surprising). Other solutions involved heavy optimisation of Ada runtime libraries and use of implementation-specific pragmata; others, for whom performance was not nearly as critical as "hard" real-time, found the Ada models and implementations sufficient and appropriate ... and were not interested in DARK at all.

DARK Activity. During this period, DARK software and documentation was given freely to Acceptor Sites upon signing a Technology Exchange Agreement (TEA); however, the SEI *did not "track" which Acceptor Site was using DARK for what purpose.* While the TEA did not require "installation" of the code, some Acceptor Sites did, in fact, install it and began porting activities of their own. Other Acceptor Sites "installed" the concepts and abstractions and their implementation, by doing textual evaluations of DARK products. DARK Project members worked with individual Acceptor Sites answering questions on an as-needed basis in an uncoordinated manner. Also during this period, the 68020-targeted version of the Kernel was ported to VAX/VMS as proof of portability.

5. *Adoption.*

Many organisations are trying to standardise on a single (family of) kernel(s), by: building their own; buying another and tailoring it to their own application; building their own hardware/software boxes as a "system solution"; enhancing an existing kernel, etc. Some organisations are doing inter-divisional work to leverage off each other's experience.

DARK Activity. Based on comments received from External Contacts, the DARK Kernel and its documentation were being used to: validate their own project work; be the target of run-time replacement tool; serve as a temporary executive until the real one under development was ready to be used (these are examples only; see also comments in Section 2.3). The biggest limitation to a wider adoption of DARK as a specific and final solution was the lack of *maintenance and enhancemement support* for it; this issue was raised time and time again.

6. *Institutionalisation.*

Some organisations are dealing with such issues at a corporate-wide, *institutional* level, as opposed to each one trying to develop a new kernel each time one would be needed. Some organisations have begun corporate-wide working groups dealing with the issues; coordination of solutions and sharing ideas/strategies became common in those organisations.

DARK Activity. Based on comments received from External Contacts, Acceptor Sites were choosing to use DARK to validate their own work; DARK models and approach fed into solutions at the corporate-wide level; (these are examples only; see also comments in Section 2.3). Again, the lack of support for DARK precluded a wider use of DARK specifically.

7. *Internalisation.* There are a set of models and approaches traditionally used by people building real-time, distributed systems. These are proven and well-understood, and are themselves the preferred (and proven) way of doing business for many real-time system builders.

DARK Activity. The models and approach underlying DARK were derived from our experience and that of people building real-time, distributed systems. In that sense, the models and approach captured by DARK were already the preferred way of doing business for a large number of systems builders. What the DARK activity provided was, in many cases, a "legitimisation" of using those well-understood models with Ada.

During the *context analysis and improvement* phase, the DARK Project was able to provide general information about the problem domain and one particular solution, the DARK Kernel. A number of presentations and road shows were given, to raise the awareness that the DARK Kernel is nothing more than one of many solutions to a generic problem.

During the *technology tailoring and installation* phase, the DARK Project was able to "throw the software and documentation over the wall," and provide minimal support, such as help with installation, bug tracking, getting Acceptor Sites to talk with each other. The software and documentation provided served as a vehicle for others to validate their work, gaining some degree of leverage.

During the *institutionalising and continuing improvement* phase, the DARK Project will have minimal involvement; consulting at best. DARK/SEI is not going to be able to "force" this stage of DARK usage, but rather plant seeds, provide an artifact and documentation that can be used to provide significant leverage, and nurture the ensuing transition activities (e.g., as a resource) as much as possible. This is where *commercialisation* would give a boost to DARK as a *product*, versus just the concepts and approach.

2.3. DARK Acceptor Sites

Version 2.0 of the DARK Kernel (and its documentation suite) was distributed to nearly 20 Acceptor Sites during 1989; even more sets of Version 2.0 documentation alone were sent to requesting individuals and institutions. All of the 350 people on the DARK External Contact list received at least one DARK-related document (each request for documentation caused the requestor to be placed on the External Contact list automatically).

The feedback to date has identified several key issues:[6]

- They [DARK users] are not using the artifact itself, but the experience that developed the Kernel. There does not seem to be a large cadre of real-time experience in the industry. What Acceptor Sites are doing is *reusing* requirements, design, and then code in that order. [Lockheed]
- The quality and focus of the documentation is outstanding. [Smiths Industries, Boeing, Lockheed]
- The Kernel code is clean, concise, well-documented. [Lockheed]
- The initial cost is null. [General Dynamics]
- DARK was interesting to me because of the lack of maintenance and enhancement support ... giv[ing] me the control and modifiability that I need to get the job done across an entire domain of projects. {Lockheed]
- The Kernel provides a good Ada-based abstraction for models we have found effective in real applications before. [Logica, Lockheed]

[6]It should be noted that not all feedback about the DARK models and abstractions has been positive. There are organisations within DoD and industry that have expressed deep concern that the DARK Project, and thus the SEI, is promulgating a "non-standard Ada" way of building systems. While that view is respected, it must be noted again that the DARK Project and DARK Transition Project tasks chose not to address that issue. In fact, those organisations that have voiced those concerns elected *not* to be Acceptor Sites.

- The use of a uniform communications mechanism ... is ... fundamental for distributed systems, where the allocation of processes to processors may be altered after the initial coding and testing. [Logica]
- We are looking at the Kernel as an example – without having to sign non-disclosure agreements – for: our own development, understanding kernel concepts in general [Ferranti, General Dynamics, Lockheed, Martin Marietta], writing requirements for procurement of a kernel [Raytheon], comparing with internal/external products [General Dynamics, Lockheed, Raytheon, Martin Marietta] etc.
- Why were they [everyone] writing them [real-time OS] when good COTS products already exist? I suspect that ... it was because their requirements were met at a 90% level by a COTS product ... [any commercial vendor] was not interested in customising their product for [everyone] ... the economic consideration is not large enough. [Lockheed]
- KFD has been used as a check list to ensure there's nothing we've not already considered in terms of our specification. [General Dynamics, Martin Marietta]
- Canabalising DARK would save on development costs. [General Dynamics]
- Transporting DARK would demonstrate reusability and transportability which in turn supports our customers concerns that the software be adaptable, maintainable, and supportable. [General Dynamics]
- To accept DARK, need proof that the functionality provided helps in producing good quality designs with adequate runtime efficiency. [Ferranti]
- The commercialisation of DARK was doomed to failure because the real-time world does not have enough commonality to make efficient runtime systems available for the general customer. [Lockheed, Smiths]
- Conflicting with (versus influencing) internal development activities. [Ferranti, Hughes]
- Using the Kernel:
    - To demonstrate our "reverse engineering" tool and to facilitate real-time system design using our tool. [CADRE]
    - As a target for ARTE replacement tool research. [Lockheed]
    - As one portion of a programme to explore scheduling algorithms and fault-tolerant, distributed, real-time kernels. [Information Systems, University in Madrid]
    - As one artifact in a real-time operating systems course for software engineering curriculum. [UHCL]
    - As a foundation for the inter-process and inter-processor communication: between various processes of a C3I system [Ford, Martin-Marietta], for a network operating system for avionics [Lockheed], for distributed simulation applications [Reflectone], for large, real-time, radar applications [TRW].

3. Objectives

Now that Kernel development is completed; the DARK Transition Project will continue the transition of models and approach taken by the DARK Project and by other kernel work.

3.1. Goals and Strategies

What *change* is the DARK Transition Project trying to effect?

- To enable more people to write real-time systems in Ada;
- To gain widespread acceptance of the view that it is acceptable to use familiar paradigms;
- To make a coherent, alternative model to Ada tasking both for distributed processing and inter-process synchronisation available to real-time system builders;
- To make a coherent, alternative model to Ada relative and absolute time available to real-time system builders;
- To get more people to accept that solutions should be based on system requirements instead of tools used to implement the solution (including a programming language);
- To return control to the user.

What *information* is the DARK Transition Project trying to obtain?

- Capture the experience of others (labs, real systems, education) at specifying and building real-time kernels;
- Capture the DARK experience;
- Codify the specifying and building of real-time kernels (this is the value-added the SEI can try to provide).

What is the *timeframe* for the DARK Transition Project?

- The DARK Transition Project has budgeted the equivalent of approximately two full-time staff members until 31 July 1990, at which point reevaluation of the DARK Transition Project is planned.
- If the planning meeting at June '90 workshop determines that it is appropriate to terminate the DARK Transition Project, that will be done, and the SEI will just respond to requests for information. However, if that meeting determines that it is premature to terminate it, the DARK Transition Project will be continued as decided at the workshop but subject to a maximum extension of six months. On termination of the DARK Transition Project, its efficacy will be re-evaluated.

Goals

To this end, the goals of the DARK Transition Project are to:

[G1]	Get the word out; *sell* DARK concepts and approach in specific; make people aware of issues of distributed, real-time Ada and kernels in general.
[G2]	Act as a *facilitator* of DARK concepts and approach and issues of distributed, real-time Ada and kernels.

[G3]	*Identify opportunities* – where related work is being done, and bring DARK to them, as one possible solution.
[G4]	Sell DARK concepts and approach *internally* within the SEI.

The criteria to determine whether or not these goals have been met are provided in Section 3.3.

Strategy

The strategy to achieve all this is to:

[S1]	*Influence* as much as possible some of the significant people dealing with distributed, real-time issues.
[S2]	Help the people who are charged with building real-time systems become *aware* of DARK concepts and approach and issues of distributed, real-time Ada and kernels.
[S3]	*Recycle* information received from others; disseminate new information developed/discovered.
[S4]	Provide *value-added* to information from outside (i.e., integrate information).
[S5]	Offer to *consult* – attend internal reviews, provide an independent, knowledgeable critique, give "guest lectures" on real-time kernels and one specific solution (DARK).
[S6]	Assess and provide requisite packaging for DARK concepts and approach for use by other Software System Programme (SSP) projects, the Education Programme, etc., internally and for self-sustaining use outside the SEI.[7]

3.2. Target Audience

Audience

Assertions:

- The DARK Transition Project is *not* out to push the entire world into understanding of real-time, embedded, distributed issues; that is a much wider activity than one to be addressed under the name "DARK."
- The DARK Transition Project is *not* out to convince the "Ada-bigots" of the world that a non-tasking solution is an acceptable solution.
- If the Ada tasking *model* is appropriate for an application, then a DARK-like solution may not be needed.
- If the Ada tasking *implementation* is sufficient for an application, then a DARK-like solution probably will not be needed.
- If the Ada distribution *model* is appropriate for an application, then a DARK-like solution may not be needed.[8]
- If the Ada distribution *implementation* is appropriate for an application, then a DARK-like solution probably will not be needed.

[7]Note: This is not trying to assert that DARK is "the" way to go, but rather one of a set of possible solutions to the generic issue of real-time, distributed Ada.

[8]Note that Ada '83 does *not* provide primitive support for distribution.

- The DARK Transition Project *is* working with that partial audience who has had *contact* with relevant issues, has *awareness* of them at some level of fluency, and is willing to (or needs to) consider the DARK models and approach among the tools they use to build systems.

The potential audience for DARK concepts, abstractions, code, and documentation includes:

[A1] **Government** – to raise their awareness of distributed, real-time Ada and kernel issues in specific, using DARK as one example of a solution; to demonstrate that the SEI knows something about such issues and, as such, is a resource to be exploited for requirements setting and review; to provide them with training materials to raise awareness of these issues within own organisation.

[A2] **Industry** – to raise their awareness of distributed, real-time Ada and kernel issues in specific, using DARK as one example of a solution; to demonstrate that the SEI knows something about such issues and, as such, is a resource to be exploited for satisfying requirements; to provide them with training materials to raise awareness of these issues within own organisation.

[A3] **Academia** – to support them in understanding some real-world issues such as distribution, real-time, Ada, kernels; to provide them with an artifact they can use to demonstrate one solution to these issues; to provide them with educational materials to explore this issue.

For each of the above, there are two sub-groups:

[E] **Experienced** – those who have built real-time, distributed systems before, whether in Ada or not, and understand the requirements that *real-time* and *distribution* place on the resultant software;

[I] **Inexperienced** – those who are yet to (successfully) build real-time, distributed systems, and are dealing with those issues for the first time.

For [E] audiences, the best use of the Kernel and its documentation is to demonstrate, openly and uncluttered by non-disclosure agreements, a set of concepts and an approach to defining real-time, distributed Kernel requirements, as well as providing one example implementation, in Ada. These can then be used (and have been used) as a "sanity check" by each sub-group against their system requirements or actual implementation. The code, if used, could be ported to another target, or used on a self-targeted system, as a component to facilitate testing until the "real thing" is built, and then the DARK Kernel may be replaced by this other, hand-crafted kernel. This sub-group of each audience needs little SEI-directed assistance.

For [I] audiences, the best use of the Kernel is to provide a model for a solution, a sample interface to model, a sample implementation, a basis for discussion of tradeoffs, etc. This sub-group of each audience may require a good deal of SEI-directed assistance.

Customer/SEI Relationships

When it is determined that a closer level of cooperation is indicated between a DARK user and the SEI, the following steps should be followed:

1. A DARK user indicates interest in doing some cooperative work with the SEI.
2. There is an agreement-in-principle (at least) for the organisation to fund such an effort, both their personnel and SEI staff required.

3. An initial draft "task plan" is proposed by or solicited from the DARK user, indicating the user's milestones, resources, and constraints. Specific tasks requiring SEI involvement are identified. This plan is reviewed with the SEI (DARK transition activity leader, other people interested in working the task, their management, the appropriate support people (e.g., facilities, TO&P monitors). Assignment of responsibility is outlined. (Iterate as required.)
4. A face-to-face meeting is set up to draft an initial technical statement of work (SOW), identify resources to be committed, timelines, responsibilities, and milestones/deliverables to be met. *Deliverables must include at least one or more submission to notable journals for publication and responses to call for papers at reputable conferences, as well as papers/presentations as appropriate within the sponsoring organisation, user group meetings, etc.*
5. Officially sign off on the SOW - both within the SEI and with the originating organisation, committing to the availability of matrixed resources and to the accomplishment of the tasks by both parties.[9]

Two specific deliverables to be sought are:

- A DARK-based *real* application.[10]
- A discussion of whether or not DARK is a *process* as well as a *product* technology, as entities that require both product and process changes are much more difficult to transition.

Thus, the DARK Transition Project is looking for "strategic clients" who are willing to:[11]

- Work with the SEI to define common problems and objectives within a specific domain context – in this case, the domain of real-time, distributed systems implemented in Ada – and to explore specific solutions within this domain context that may be used – in this case, the artifacts produced by the DARK Project;
- Cooperate with the SEI in the definition of new engineering solutions to the software problems confronting the client – in this case, evaluating the Kernel and its documentation and using them;
- Accept close cooperation with the SEI in the use of technology – in this case, providing information exchange about the Kernel and its documentation in specific and the technology areas of distributed, real-time systems in general;
- Work with the SEI to disseminate the results to other clients with similar problems – in this case, supporting the electronic exchange for DARK users if possible, co-authoring papers and presentations with SEI staff and submitting them to reputable journals and conferences, participating in SEI-sponsored workshop;
- Share a funding burden for the development and installation of the technology – provide money and/or hardware resources for tasks being performed jointly between the SEI and an organisation.

[9]This official agreement is intended to be more binding on all signatories than the earlier TEA. The TEA levied minimal requirements on the Acceptor Sites – and that is just what the SEI received in return. It should also be noted that the SEI was reticent in requesting information it had promised to request.

[10]Ferranti may be a good bet for this, as they have a small application they will themselves be using to test their port.

[11]Adapted from *Technology Applications Plan*, DRAFT Version 2.6, 8 August 1989.

The details of the relationship between the SEI and a strategic client would be worked out on a case-by-case basis, and these activities would be run as separate projects. See (4b) on Page 26.

This will be determined via the project review process.

3.3. Success Criteria

These are the criteria used to measure the success of the goals for DARK transition, stated in Section 3.1.

[C1] *Selling DARK:*

- Giving presentations to industry, academia, government;
- Giving tutorials to above;
- Giving papers, presentations, tutorials at relevant and visible conferences;
- Having others write about DARK;
- Sponsoring a workshop with published proceedings;
- Using Kernel models and concepts: in developmental products themselves; as criteria against which others are compared and tradeoffs discussed (e.g., as basis for RFP or analysis); as a framework for discussion of general issues;
- Getting invited to conferences and meetings as well as applying and being accepted;
- Having DARK Technical Reports and papers cited;
- People requesting or obtaining DARK Technical Reports, papers, code;
- Having people *come to us.*

[C2] *Facilitating technology transition/exploration/adoption:*

- Creating an electronic exchange for DARK external contacts;
- Getting DARK users talking with each other;
- Getting people dealing with the wider issues of real-time, distribution, and Ada talking with each other;
- Being recognised as a source of relevant information (whether provided directly or indirectly via a goto-statement);
- Getting DoD programme offices to come to the SEI for expertise and references;
- Holding credible, high-powered, well-respected workshops;
- Becoming recognised such that government, industry, and academia want to send their people here as affiliates, as students, or to workshops, etc;
- Making DARK available as a testbed in which to explore other

real issues for real-time systems, such as: fault tolerance, reliability, multi-level security, performance, etc.

[C3] *Identify opportunities:*

- Having a reputation, as a Project, as a part of the SEI, to be considered credible in a "new" environment;
- Being called upon to provide inputs into as-yet untapped work.

[C4] Sell DARK *internally*:

- Having other SEI projects and programmes look at DARK as a candidate for use, for inclusion in a toolkit, as relevant for citing, as appropriate for referals of their clients, etc.

4. Top-Level Task Description

4.1. Tasks

Each of the following seven tasks is described in the following manner:

1. Short task description

   a. One or more sub-task descriptions for each task description; activity or activities to be performed in order to complete the task.

   Priority given to each sub-task description; low, medium, high.

   Completed by for each sub-task; scheduled completion date for primary actions; some low-level, on-going work may be incurred.

   Mapping to goals (of the form [G*], as defined in Section 3.1), strategies (of the form [S*], as defined in Section 3.1), audience (of the form [A*], as defined in Section 3.2), and success criteria (of the form [C*], as defined in Section 3.3).

1. Establish "final"/Version 3.0 baseline:

   a. Determine status of current External Contacts: what they have, what they are using, to what extent are they using it, in what application areas, on what host/target systems, etc.

   Priority: high

   Completed by: 31 January 1990

   b. Create and distribute final documentation and software package – Version 3.0.

   Priority: high

   Completed by: 31 January 1990 (on-going distribution if needed)

   c. Investigate making all DARK documentation available electronically (e.g., "automatic" translation of text and figures into Interleaf, creating an integrated document, and then making a *postscript* version thereof available).

   Priority: low

   Completed by: Spring 1990 (go/no-go decision)

   [G*] [S2] [S6] [A*] [C1] [C4]

2. Create "clearinghouse" for the above information, bug reports, inter-site communication, etc.

   a. Set up electronic distribution mechanisms for source code, Kernel documentation, and other DARK-related papers.

   Priority: high

   Completed by: 31 January 1990

   b. Create electronic mailing lists/bboards.

   Priority: high

Completed by: 31 January 1990

c. Continue to support manual distribution to support those electronically disenfranchised.

Priority: high

Completed by: On-going 1990 (as needed)

d. Use DARK UPDATE newsletter as the announcement vehicle for the electronic availability of DARK, the mailing lists, and other DARK transition activities, as soon as Version 3.0 available.

Priority: high

Completed by: 31 January 1990

e. Collect information about optimisations, extensions, and enhancements to the Kernel that could be used in a follow-on project based on DARK or a commercialisation effort.

Priority: medium

Completed by: On-going 1990

[G*] [S2] [S3] [A*] [C2]

3. Short turn-around "advertisement" of DARK

a. DARK articles in trade journals

Priority: medium

Completed by: mid-February 1990

[G1] [S2] [A*] [C1]

4. Support existing external usage of DARK artifacts

a. Be available to answer questions

Priority: high

Completed by: On-going 1990

b. Determine whether closer level of cooperation is indicated between a DARK user and the SEI. Steps to be followed are described in Section 3.2.

Priority: high

Completed by: 31 July 1990 (each task will have a life of its own, coordinated, to the degree appropriate, through the DARK Transition Project with other related work; formal coordination may end 31 July 1990, but the tasks themselves may continue)

c. Track ongoing DARK-related activities at some coherent level (e.g., update DARK usage database as appropriate with new contacts, new ports, etc; log/document key contacts; write up key issues and news and make them available electronically).

Priority: high

Completed by: 31 January 1990 (initial entry; on-going thereafter)

d. Continue newsletter for 6 - 18 months, with discussions of upcoming events of interest, advertisements of what is being done (new products/tutorials/presentations/contacts); featuring "guest articles" from people using DARK or dealing with distributed, real-time, Ada kernel issues in general. Ensure this is made available electronically.

Priority: high / medium

Completed by: 31 January 1990 (next issue; as appropriate thereafter)

[G2] [G3] [S5] [A2] ([A1] [A3] if come to life) [C2] [C3]

5. Creating a portable demonstration.

   a. Creating a portable demonstration – work with CADRE (*actively!*). We need to reactivate the relationship with CADRE and establish a more formal agreement, including co-authoring a paper/presentation.

   Priority: low

   Completed by: 31 January 1990 (re-initiate contact; revisit SOW)

[G1] (could lead to [G2], [G3]) [S2] [S4] [A*] [C1]

6. Plan and hold a workshop in June 1990 – *Distributed Real-Time Ada Workshop.*

   a. It is planned to be three-pronged, accommodating government, academia, and industry audiences. It is anticipated to be three days long, with presentations solicited on the following areas:

      - One half-day: Government requirements and activities.
      - One half-day: Academia requirements and activities.
      - One half-day: Industry requirements and activities.
      - 4th half-day: Common issues and directions among Government, Academia, and Industry.
      - 5th half-day: Focus specifically on who is doing what with DARK.
      - 6th half-day: DARK transition-specific planning with active DARK users and key people involved with real-time, distributed systems issues.

   The workship will be mostly concerned with discussing real experiences, tools and how they were used, expositions on methodologies (what worked, didn't, what was modified to work next time), "generic/reusable" paradigms, models, and abstractions, and so forth. The position papers will be published as an SEI Technical Report.

   Priority: high

   Completed by: 31 January 1990 (announcement), June 1990 (workshop), July 1990 (proceedings)

[G*] [S2] [S3] [S4] [A*] [C2] [C3]

7. Prepare a tutorial on *Components of Real-Time Kernels.*

   a. This will be done collaboratively with DARK users and SEI Affiliates. Ideally, the tutorial will begin by looking at general system *requirements* from a user's point of view (with material provided by a DARK user); followed by a discussion of general *characteristics* of real-time kernels (with material provided by an SEI Affiliate); followed by an examination of how one such kernel – *the DARK Kernel* – addresses those characteristics (with material provided via the DARK transition activity staff); and the tutorial will conclude with a *case study* of using the DARK Kernel (with material provided by a DARK user). The resulting tutorial would, hopefully, be useful both in an industrial setting and an academic environment, and could be offered as a portion of an existing SEI Curriculum Module.

Potential audiences include those at: TRI-Ada '90, the SEI Affiliates Symposium (both requiring near-term commitments), Affiliates organisations (especially DARK users), DoD Programme Offices (especially those grappling with the issues of real-time, distribution, and Ada), CSC/SIGCSE '91, perhaps some NSIA/AIAA/EIA software conferences(?).

Consider video-taping one of these presentations (but do *not* produce a second set of materials specifically for a video presentation; they need to get done right – the first time).

Priority: medium

Completed by: 30 April 1990 (top-level outline), 30 June 1990 (first draft), 31 July 1990 (final copy)

[G1] [G2] [S1] [S2] [S3] [S4] [S6] [A*] [C1] [C2]

4.2. Relationship of the Phase Approach to the DARK Transition Tasks

This section provides a simple mapping from the task descriptions in Section 4.1 to the phases of transition, as described in Chapter 2.

1. *Contact:* (3a) (5a)
2. *Awareness:* (3a) (4d) (5a) (7a)
3. *Understanding:* (2d) (4a) (4c) (4d) (5a) (6a) (7a)
4. *Installation:* (1b) (1c) (2a) (2b) (2c) (2d) (4a) (4c) (4d) (6a)
5. *Adoption:* (2e) (4a) (4b) (4d) (6a)
6. *Institutionalisation:* (4b)
7. *Internalisation:*

Miscellaneous (clerical): (1a) (4c)

5. Bibliography

[1] Ada Runtime Environment Working Group.
A Canonical Model and Taxonomy of Ada Runtime Environments.
Technical Report, SIGAda, November, 1986.

[2] Ada Runtime Environment Working Group.
First Annual Survey of Mission Critical Application Requirements.
Technical Report Release 1.0, SIGAda, November, 1986.

[3] Firth, R.
A Pragmatic Approach to Ada Insertion.
In *Proceedings of the International Workshop on Real-Time Ada Issues,* pages 24-26. May, 1987.

[4] Bamberger, J., C. Colket, R. Firth, D. Klein, R. Van Scoy.
Kernel Facilities Definition.
Technical Report CMU/SEI-88-TR-16, ESD-TR-88-17, ADA198933, Software Engineering Institute, December, 1989.